Homework 4: Blackjack

Course: CS 221 Spring 2019
Name: Bryan Yaggi

The search algorithms explored in the previous assignment work great when you know exactly the results of your actions. Unfortunately, the real world is not so predictable. One of the key aspects of an effective AI is the ability to reason in the face of uncertainty.

Markov decision processes (MDPs) can be used to formalize uncertain situations. In this homework, you will implement algorithms to find the optimal policy in these situations. You will then formalize a modified version of Blackjack as an MDP, and apply your algorithm to find the optimal policy.

Problem 1: Value Iteration

In this problem, you will perform the value iteration updates manually on a very basic game just to solidify your intuitions about solving MDPs. The set of possible states in this game is $\{-2, -1, 0, 1, 2\}$. You start at state 0, and if you reach either -2 or 2, the game ends. At each state, you can take one of two actions: $\{-1, +1\}$.

If you're in state s and choose -1:

- You have an 80% chance of reaching the state $s - 1$.
- You have a 20% chance of reaching the state $s + 1$.

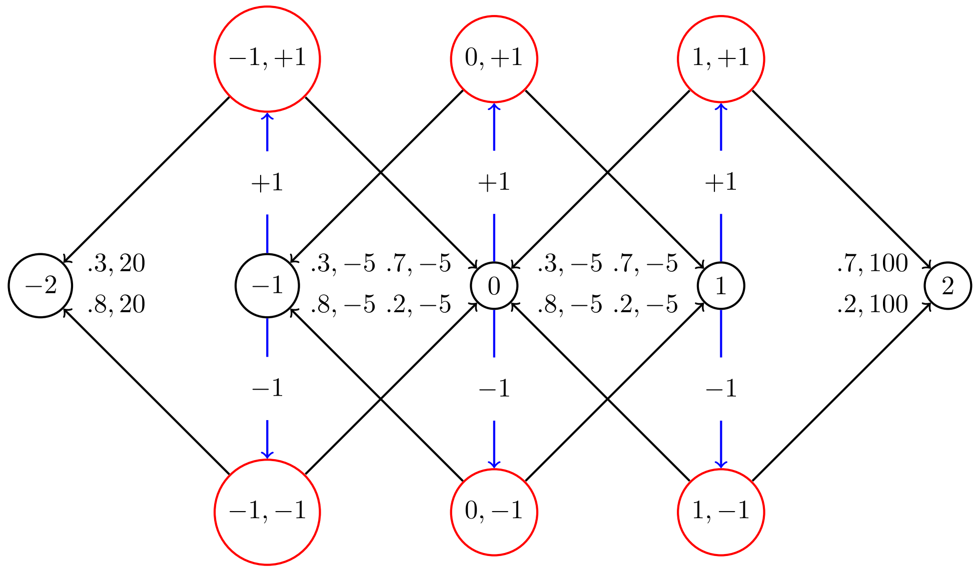
If you're in state s and choose $+1$:

- You have an 70% chance of reaching the state $s + 1$.
- You have a 30% chance of reaching the state $s - 1$.

If your action results in transitioning to state -2, then you receive a reward of 20. If your action results in transitioning to state 2, then your reward is 100. Otherwise, your reward is -5. Assume the discount factor γ is 1.

(a) Give the value of $V_{opt}(s)$ for each state s after 0, 1, and 2 iterations of value iteration. Iteration 0 just initializes all the values of V to 0. Terminal states do not have any optimal policies and take on a value of 0.

(b) What is the resulting optimal policy π_{opt} for all non-terminal states?



$$V_{opt}^{(t)}(s) = max_{a \in actions(s)} \sum_{s'} T(s,a,s')[reward(s,a,s') + \gamma V_{opt}^{(t-1)}(s')]$$

Iteration 0:

s	-2	-1	0	1	2
V_{opt}	0	0	0	0	0
π_{opt}	*none*	*none*	*none*	*none*	*none*

$$V_{opt}^{(1)}(-1) = max(.7(-5 + 1(0)) + .3(20 + 1(0)), .8(20 + 1(0)) + .2(-5 + 1(0))) = 15$$
$$V_{opt}^{(1)}(0) = max(.7(-5 + 1(0)) + .3(-5 + 1(0)), .8(-5 + 1(0)) + .2(-5 + 1(0))) = -5$$
$$V_{opt}^{(1)}(1) = max(.7(100 + 1(0)) + .3(-5 + 1(0)), .8(-5 + 1(0)) + .2(100 + 1(0))) = 68.5$$

Iteration 1:

s	-2	-1	0	1	2
V_{opt}	0	15.0	-5.0	68.5	0
π_{opt}	*none*	-1	*either*	$+1$	*none*

$$V_{opt}^{(2)}(-1) = max(.7(-5 + 1(-5)) + .3(20 + 1(0)), .8(20 + 1(0)) + .2(-5 + 1(-5))) = 14$$
$$V_{opt}^{(2)}(0) = max(.7(-5 + 1(68.5)) + .3(-5 + 1(15)), .8(-5 + 1(15)) + .2(-5 + 1(68.5))) = 47.45$$
$$V_{opt}^{(2)}(1) = max(.7(100 + 1(0)) + .3(-5 + 1(-5)), .8(-5 + 1(-5)) + .2(100 + 1(0))) = 67$$

Iteration 2:

s	-2	-1	0	1	2
V_{opt}	0	14.0	47.45	67.0	0
π_{opt}	*none*	$+1$	$+1$	$+1$	*none*

Problem 2: Transforming MDPs

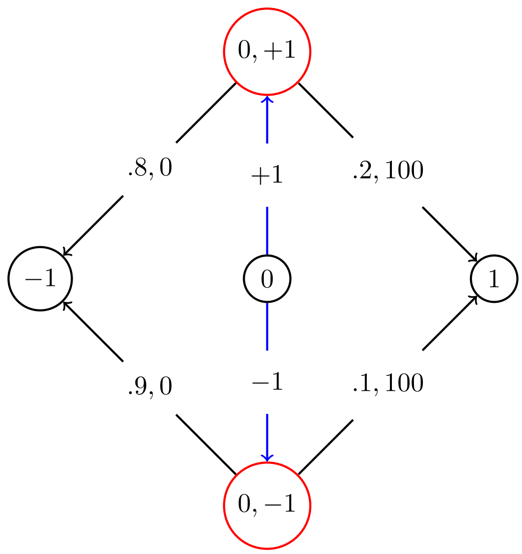
Let's implement value iteration to compute the optimal policy on an arbitrary MDP. Later, we'll create the specific MDP for Blackjack.

(a) If we add noise to the transitions of an MDP, does the optimal value always get worse? Specifically, consider an MDP with reward function $Reward(s, a, s')$, states $States$, and transition function $T(s, a, s)$. Let's define a new MDP which is identical to the original, except that on each action, with probability $\frac{1}{2}$, we randomly jump to one of the states that we could have reached before with positive probability. Formally, this modified transition function is:

$$T'(s, a, s') = \frac{1}{2}T(s, a, s') + \frac{1}{2}\frac{1}{|\{s'' : T(s, a, s'') > 0\}|}$$

Let V_1 be the optimal value function for the original MDP, and V_2 the optimal value function for the modified MDP. Is it always the case that $V_1(s_{start}) \geq V_2(s_{start})$? If so, prove it on the written portion and put return None for each of the code blocks. Otherwise, construct a counter example by filling out `CounterexampleMDP` in `submission.py`.

This is not always the case. Here is a counterexample.



s	-1	0	1
V_1	0	20	0
V_2	0	35	0

(b) Suppose we have an acyclic MDP for which we want to find the optimal value at each node. We could run value iteration, which would require multiple iterations – but it would be nice to be more efficient for MDPs with this acyclic property. Briefly explain an algorithm that will allow us to compute V_{opt} for each node with only a single pass over all the (s, a, s') triples.

The alogrithm would involve calculating V_{opt} for the end states and working backwards to the start state.

(c) Suppose we have an MDP with states $States$ and a discount factor $\gamma < 1$, but we have an MDP solver that can only solve MDPs with discount factor of 1. How can we leverage the MDP solver to solve the original MDP?

Let us define a new MDP with states $States' = States \cup \{o\}$, where o is a new state. Let's use the same actions $Actions'(s) = Actions(s)$, but we need to keep the discount $\gamma' = 1$. Your job is to define new transition probabilities $T'(s, a, s')$ and rewards $Reward'(s, a, s')$ in terms of the old MDP such that the optimal values $V_{opt}(s)$ for all $s \in States$ are equal under the original MDP and the new MDP.

Hint: If you're not sure how to approach this problem, go back to the notes from the first MDP lecture and read closely the slides on convergence, toward the end of the deck.

From the notes: "We can reinterpret the discount $\gamma < 1$ condition as introducing a new transition from each state to a special end state with probability $(1 - \gamma)$, multiplying all the other transition probabilities by γ, and setting the discount to 1. The interpretation is that with probability $1 - \gamma$, the MDP terminates at any state."

$$T'(s, a, s') = \gamma T(s, a, s')$$
$$Rewards'(s, a, s') = Rewards(s, a, s')$$